

- 1 -

PERCEPTION-AWARE LOW-POWER AUDIO DECODER FOR PORTABLE DEVICES

Field of the Invention

The present invention relates generally to low-power decoding in multimedia applications and, in particular, to a method and apparatus for decoding audio data, and to a computer program product including a computer readable medium having recorded thereon a computer program for decoding audio data.

Background

Increasingly, many portable consumer electronics devices, such as mobile phones, portable digital assistants (PDA) and portable audio players comprise embedded computer systems. These embedded computer systems are typically configured according to general-purpose computer hardware platforms or architecture templates. The only difference between these consumer electronic devices is typically the software application that is being executed on the particular device. Further, several different functionalities are increasingly being clubbed into one device. For example, some mobile phones also work as portable digital assistants (PDA) and/or portable audio players. Accordingly, there has been a shift of focus in the portable embedded computer systems domain towards appropriate software-implementations of different functionalities, rather than tailor-made hardware for different applications.

Power consumption of the computer systems embedded in the portable devices is probably the most critical constraint in the design of both, hardware and software, for such portable devices. One known method of minimising power consumption of computer systems embedded in portable devices is to dynamically scale the voltage and frequency (i.e., clock frequency) of the processor of an embedded computer system in response to the variable workload involved in processing multimedia streams.

- 2 -

Another known method of minimising power consumption of computer systems embedded in portable devices uses buffers to smooth out multimedia streams and decouple two architectural components having different processing rates. This enables the embedded processor to be periodically switched off or for the processor to be run at a lower frequency, thereby saving energy. There are also a number of known scheduling methods addressed at the problem of maintaining a Quality-of-Service (QoS) requirement associated with multimedia applications and at the same time minimizing power consumption of an embedded computer system.

Summary

It is an object of the present invention to substantially overcome, or at least ameliorate, one or more disadvantages of existing arrangements.

According to one aspect of the present invention there is provided a method of decoding audio data representing an audio clip, said method comprising the steps of:

selecting one of a predetermined number of frequency bands;

decoding a portion of the audio data representing said audio clip according to the selected frequency band, wherein a remaining portion of the audio data representing said audio clip is discarded; and

converting the decoded portion of audio data into sample data representing the decoded audio data.

According to another aspect of the present invention there is provided a decoder for decoding audio data representing an audio clip, said method comprising the steps of:

decoding level selection means for selecting one of a predetermined number of frequency bands;

decoding means for decoding a portion of the audio data representing said audio clip according to the selected frequency band, wherein a remaining portion of the audio data representing said audio clip is discarded; and

- 3 -

data conversion means for converting the decoded portion of audio data into sample data representing the decoded audio data.

According to still another aspect of the present invention there is provided a portable electronic device comprising:

5 decoding level selection means for selecting one of a predetermined number of frequency bands;

decoding means for decoding a portion of audio data representing an audio clip according to the selected frequency band, wherein a remaining portion of the audio data representing said audio clip is discarded; and

10 data conversion means for converting the decoded portion of audio data into sample data representing the decoded audio data.

Other aspects of the invention are also disclosed.

Brief Description of the Drawings

One or more embodiments of the present invention will now be described with
15 reference to the drawings and appendices, in which:

Fig. 1 is a schematic block diagram of a portable computing device comprising a processor, upon which embodiments described can be practiced;

Fig. 2 shows the processor of Fig. 1 taking a coded bitstream as input and producing a stream of decoded pulse code modulated (PCM) samples;

20 Fig. 3 shows the frame structure of an MPEG 1, Layer 3 (i.e., MP3) standard bitstream;

Fig. 4 is a block diagram showing the modules of a standard MP3 decoder together with the proposed new decoder architecture;

Fig. 5 shows an internal buffer and playout buffer used by the processor of Fig. 1 in
25 decoding audio data;

- 4 -

Fig. 6 is a graph showing the cycle requirement for the processor of Fig. 1 per granule, corresponding to an audio clip, for a predetermined duration;

Fig. 7 shows the processor cycles required within any interval of length t corresponding to the decoding levels of the preferred embodiment; and

5 Fig. 8 shows a method of decoding audio data in the form of a coded bit stream, in accordance with the preferred embodiment.

Detailed Description including Best Mode

Where reference is made in any one or more of the accompanying drawings to steps and/or features, which have the same reference numerals, those steps and/or features have
10 for the purposes of this description the same function(s) or operation(s), unless the contrary intention appears.

It is to be noted that the discussions contained in the "Background" section and that above relating to prior art arrangements relate to discussions of documents or devices which form public knowledge through their respective publication and/or use. Such
15 should not be interpreted as a representation by the present inventor(s) or patent applicant that such documents or devices in any way form part of the common general knowledge in the art.

Most perceptual audio coder/decoders (i.e., codecs) are designed to achieve transparent audio quality at least at high bit rates. The frequency range of a high quality
20 audio codec such as MP3 is up to about 20 kHz. However, most adults, particularly older ones, can hardly hear frequency components above 16 kHz. Therefore, it is unnecessary to determine the perceptually irrelevant frequency components. Further, within the wide swath of frequencies that most people can hear, some bands register more loudly than others. In general, the high frequency bands are perceptually less important than the low
25 frequency bands. There is little perceptual degradation if some high frequency components are left un-decoded. A standard decoder such as an MP3 decoder will simply

decode everything in an input bit stream without considering the hearing ability of individual users with or without hearing loss. This results in a significant amount of irrelevant computation, thereby wasting battery power of a portable computing device or the like using such a decoder.

5 A method 800 of decoding audio data in the form of a coded bit stream, in accordance with the preferred embodiment, is described below with reference to Figs. 1 to 8. The principles of the preferred method 800 described herein have general applicability to most existing audio formats. However, for ease of explanation, the steps of the preferred method 800 are described with reference to the MPEG 1, Layer 3 audio format, 10 also known as MP3, audio format. MP3 is a non-scalable codec and has widespread popularity. The method 800 is particularly applicable to non-scalable codecs like MP3 and also Advanced Audio Coding (AAC). Non-scalable codecs incur a lower workload and are more popular than scalable codecs, such as an MPEG-4 scalable codec, where only a base layer is typically decoded with an enhancement layer being ignored.

15 The method 800 integrates an individual user's own judgment on the desired audio quality allowing a user to switch between multiple output quality levels. Each such level is associated with a different level of power consumption, and hence battery lifetime. The described method 800 is perception-aware, in the sense that the difference in the perceived output quality associated with the different levels is relatively small. But 20 decoding the same audio data, such as an audio clip in the form of a coded bit stream, at a lower output quality level leads to significant savings in the energy consumed by the processor embedded in a portable device.

To evaluate the perceptual quality of any audio codec, rigorous subjective listening tests are carried out. These tests are usually conducted in a quiet environment with high 25 quality headphones by expert listeners or panels without any hearing loss. However, the realistic environments for ordinary users are usually very different. Firstly, it is relatively

- 6 -

rare for a portable audio player to be used in a quiet environment, for example in the living room of one's home. It is far more common to use portable audio players on the move and in a variety of environments such as in a bus, train, or in a flight, using simple earpieces. These differences have important implications on the audio quality required.

5 According to experiments carried out by the present inventors, it is hard for most users to distinguish between Compact Disc (CD) and Frequency Modulation (FM) quality audio in a noisy environment. Most users appear to be more tolerant to a small quality degradation in such environments. The method 800 enables the user to change the decoding profile to adapt to the listening environment, while a standard MP3 decoder
10 cannot.

Different applications and signals require different bandwidth. For example, a story-telling audio clip requires significantly less bandwidth compared to a music clip. The method 800 allows the user to choose an appropriate decoding profile suitable for the particular service and signal type also prolonging the battery life of a portable computing
15 device using the method 800. The method 800 allows users to control the tradeoff between the battery life and the decoded audio quality, with the knowledge that slightly degraded audio quality (this degradation may not even be perceptible to the particular user) can significantly increase the battery life of a portable audio player, for example. This feature allows the user to tailor the acceptable quality level of the decoded audio
20 according to their hearing ability, listening environment and service type. For example, in a quiet environment the user may prefer perfect sound quality with more power consumption. On the other hand, the user might prefer a longer battery life with slightly degraded audio quality during a long haul flight.

The method 800 is preferably practiced using a battery-powered portable computing
25 device 100 (e.g., a portable audio (or multi-media) player, a mobile (multi-media) telephone, a PDA or the like) such as that shown in Fig. 1. The processes of Figs. 2 to 8

may be implemented as software, such as a software program executing within the portable computing device 100. In particular, the steps of the method 800 are effected by instructions in the software that are carried out by the portable computing device 100. The instructions may be formed as one or more software modules, each for performing
5 one or more particular tasks. The software may also be divided into two separate parts, in which a first part performs the method 800 and a second part manages a user interface between the first part and the user. The software may be stored in a computer readable medium, including the storage devices described below, for example. The software may be loaded into the portable computing device 100 by a manufacturer, for example, from
10 the computer readable medium, via a serial link and then be executed by the portable computing device 100. A computer readable medium having such software or computer program recorded on it is a computer program product. The use of the computer program product in the computer system 100 preferably effects an advantageous apparatus for implementing the described method 800.

15 The portable computing device 100 includes at least one processor unit 105, and a memory unit 106, for example formed from semiconductor random access memory (RAM) and read only memory (ROM). The portable computing device 100 may also comprise a keypad 102, a display 114 such as a liquid crystal display (LCD), a speaker 117 and a microphone 113. The portable computing device 100 is preferably powered by
20 a battery. A transceiver device 116 is used by the portable computing device 100 for communicating to and from a communications network 120 (e.g., the telecommunications network), for example, connectable via a wireless communications channel 121 or other functional medium. The components 105 to 117 of the portable computing device 100 typically communicate via an interconnected bus 104.

25 Typically, the application program is resident in ROM of the memory device 106 and is read and controlled in its execution by the processor 105. Still further, the software

can also be loaded into the portable computing device 100 from other computer readable media. The term “computer readable medium” as used herein refers to any storage or transmission medium that participates in providing instructions and/or data to the portable computing device 100 for execution and/or processing.

5 The method 800 may alternatively be implemented in dedicated hardware unit comprising one or more integrated circuits performing the functions or sub functions of the described method.

In accordance with the method 800, a decoding level selected by a user to decode any audio clip determines the frequency with which the processor 105 is to be executed.

10 In contrast to many known dynamic voltage/frequency scaling methods, the method 800 does not involve any runtime scaling of the processor 105 voltage or frequency. If the processor 105 has a fixed number of voltage-frequency operating points, the decoding levels in the method 800 may be tuned to match these operating points.

In the method 800, the frequency bandwidth of the portable computing device 100
15 comprising an audio decoder (e.g., an MP3 decoder) implemented therein, is partitioned into a number of groups that is equal to the number of decoding levels. These groups are preferably ordered according to their perceptual relevance, which will be described in detail below. If there are four levels of decoding (i.e. Levels 1–4) then the frequency bandwidth group that has the highest perceptual relevance may be associated with Level 1
20 and the group that has the lowest perceptual relevance may be associated with Level 4. Such a partitioning of the frequency bandwidth into four levels in the case of MP3 is shown in Table 1 below. Column 2 of Table 1 (i.e., Decoded subband index) is described below.

Table 1

Decoding level	Decoded subband index	Frequency range (Hz)	Perceived quality level
-----------------------	------------------------------	-----------------------------	--------------------------------

Level 1	0-7	0 – 5512.5	AM quality
Level 2	0-15	0 – 11025	Near FM quality
Level 3	0-23	0 – 16537.5	Near CD quality
Level 4	0-31	0 – 22050	CD quality

The processor 105 implementing the steps of the method 800 may be referred to as a “Perception-aware Low-power MP3 (PL-MP3)” decoder. The method 800 is not only useful with general-purpose voltage and frequency scalable processors, but also with
5 general-purpose processors without voltage and frequency scalability.

The method 800 may also be used with a processor that does not allow frequency scaling and is not powerful enough to do full MP3 decoding. In this instance, the method 800 may be used to decode regular MP3 files at a relative lower quality.

The method 800 allows a user to choose a decoding level (i.e., one of four such
10 levels) depending on processing power supplied by the processor 105. The method 800 is executed by the processor 105 based on the decoding level selected by the user. Each level is associated with a different level of power consumption and a corresponding output audio quality level. The processor 105 takes audio data in the form of a coded bit stream as input and produces a stream of decoded data in the form of pulse code
15 modulated (PCM) samples, as seen in Fig. 2. The method 800 may be applied to decode a coded bit stream that is being downloaded or streamed from a network. The method 800 may also be used to decode an audio clip in the form of a coded bit stream stored within the memory 106, for example, of the portable computing device 100.

When an audio clip in the form of a coded bit stream is decoded at Level 1, only the
20 frequency range 0 to 5512.5 Hz associated with this level is decoded. At higher levels (i.e., Level 2 to 3), a larger frequency range is decoded and finally at Level 4, the entire frequency range is decoded. Although the computational workload associated with the method 800 scales almost linearly with the decoding level, the lower frequency ranges

have a much higher perceptual relevance compared to the higher ones, as described above. Therefore, when an audio clip is decoded at a lower level, by sacrificing only a small fraction of the output quality, the processor 105 may be run at a much lower frequency (i.e., clock frequency) and voltage, when compared to a higher decoding level.

5 Recently a number of hardware implementations of audio decoders have been developed. Some of these hardware implementations include hardwired decoder chips which have been designed for very low power consumption. An example of such a decoder chip is the ultra low-power MP3 decoder from Atmel CorporationTM, which is designed especially to handle MP3 ring tones in mobile phones.

10 The method 800 lowers the power consumption of the processor 105 executing the software implementing the steps of the method 800. The method 800 does not rely on any specific hardware implementations or on any co-processors to implement specific parts of the decoder. The method 800 is very useful for use with PDAs, portable audio players or mobile phones and the like comprising powerful voltage and frequency
15 scalable processors, which may all be used as portable audio/video players.

Like many other multimedia bitstreams, the MP3 bitstream has a frame structure, as seen in Fig. 3. A frame 300 of the MP3 bitstream contains a header 301, an optional CRC 302 for error protection, a set of control bits coded as side information 303, followed by the main data 304 consisting of two granules (i.e., Granule 0 and Granule 2) which are the
20 basic coding units in MP3. For stereo audio, each granule (e.g., Granule 1) contains data for two channels, which consists of scale factors 305 and Huffman coded spectral data 306. It is also possible to have some ancillary data inserted at the end of each frame. The method 800 processes such an MP3 bit stream frame by frame or granule by granule.

The method 800 of decoding audio data will now be described with reference to
25 Fig. 8. The method 800 may be implemented as software resident in the ROM 106 and being controlled in its execution by the processor 105. The portable computing device

- 11 -

100 implementing the method 800 may be configured in accordance with a standard MP3 audio decoder 400 as seen in Fig. 4. Each of the steps of the method 800 may be implemented using separate software modules.

The method 800 begins at the first step 801, where the one of the four decoding
5 levels (i.e., Levels 1 – 4) of Table 1 are selected. For example, the user of the portable computing device 100 may select one of the four decoding levels using the keypad 102. The processor 105 may store a flag in the RAM of the memory 106 indicating which one of the four decoding levels has been selected.

At the next step 802, the processor 105 parses data in the form of a coded input bit
10 stream and stores the data in an internal buffer 500 (see Fig. 5) configured within the memory 106. The internal buffer 500 will be described in more detail below. Then at step 803, the processor 105 decodes the side information of the stored data using Huffman decoding. Step 803 may be performed using a software module such as the Huffman decoding software module 401 of the standard MP3 decoder 400, as seen in Fig. 4.

15 The method 800 continues at the next step 804, where the processor 105 converts a frequency band of the decoded audio data into PCM audio samples, according to the decoding level selected at step 801. For example, if Level 1 was selected at step 801, then the decoded audio data in the frequency range 0 to 5512.5 Hz will be converted into PCM audio samples at step 804. Step 804 may be performed by software modules such as the
20 dequantization software module 402, the inverse modified discrete cosine transform (IMDCT) software module 403 and the polyphase synthesis software module 404 of the standard MP3 decoder 400 as seen in Fig. 4.

The method 800 concludes at the next step 805, where the processor 105 writes the PCM audio samples into a playout buffer 501 (see Fig. 5) configured within memory 106.
25 This playout buffer 501 may then be read by the processor 105 at some specified rate and be output as audio via the speakers 117.

The three modules of a standard MP3 decoder 400, which incur the highest workload are the de-quantization module 402, the IMDCT module 403 and the polyphase synthesis filterbank module 404. Traditionally, the standard MP3 decoder 400 decodes the entire frequency band, which corresponds to the highest computational workload. As
5 seen in Fig. 4, in accordance with the preferred method 800, depending on the decoding level (i.e., Levels 1 to 3), the de-quantization module 402, the IMDCT module 403 and the polyphase synthesis filterbank module 403 process only a partial frequency range and thereby incur less computational cost.

There are several known optimization methods used for memory and/or
10 computationally efficient implementations such as the “Do Not Zero-Pute” algorithm described by De Smet et al in the publication entitled “*Do Not Zero-Pute: An Efficient Homespun MPEG-Audio Layer II Decoding and Optimisation Strategy*”, In Proc. Of ACM Multimedia 2004, Oct. 2004. The Do Not Zero-Pute algorithm tries to optimize the polyphase filterbank computation in the MPEG 1 layer II by eliminating costly computing
15 cycles being wasted at processing useless zero-valued data. The present inventors classify this kind of approach as eliminating redundant computation. In contrast, the method 800 partitions the workload according to frequency bands with different perceptual relevance and allows the user to eliminate the irrelevant computation.

The reduction of workload in the three computationally most demanding modules,
20 namely the de-quantization module 402, the IMDCT module 403 and the polyphase synthesis filterbank module 404, is expressed in the following Equations (1) to (4).

The computation required to be performed by the processor 105 for the de-quantization of a granule (in the case of long blocks) is expressed as Equation (1) as follows:

$$\begin{aligned}
 x_{r_i} = & \text{sign}(is_i) * |is_i|^{\frac{4}{3}} * 2^{\frac{1}{4}(\text{global_gain}[gr] - 210)} \\
 & * 2^{-(\text{scalefac_multiplier} * (\text{scalefac_l}[sfb][ch][gr] + \text{preflag}[gr] * \text{pretab}[sfb]))}
 \end{aligned}
 \tag{1}$$

where is_i is the i -th input coefficient being dequantized, $\text{sign}(is_i)$ is the sign of is_i ,
 5 global_gain is the logarithmical quantizer step size for the entire granule gr.
 Scalefac_multiplier is the multiplier for scale factorbands. Scalefac_1 is the
 logarithmically quantized factor for scale factorband sfb of channel ch of granule gr.
 Preflag is the flag for additional high frequency amplification of the quantized values.
 Pretab is the preemphasis table for scale factorbands. x_{r_i} is the i -th dequantized
 10 coefficient.

For the standard MP3 decoder 400 not executing the steps of the method 800,
 $i = 0, 1, \dots, N-1$ and $N = 576$, while $i = 0, 1, \dots, sbl * 18 - 1$ for the processor 105 of such a decoder
 400 executing the steps of the method 800. For example, the range for Level 1 is reduced
 to $i = 0, 1, \dots, 143$.

15 The computation required for the IMDCT module 403 may be expressed in
 accordance with Equation (2) as follows:

$$x_i = \sum_{k=0}^{n/2-1} X_k \cos\left(\frac{\pi}{2n}\left(2i+1+\frac{n}{2}\right)(2k+1)\right)
 \tag{2}$$

for $i = 0, 1, \dots, n-1$ and $n = 36$, where X_k is the k -th input coefficient for IMDCT operations
 and x_i is the i -th output coefficient. For the standard MP3 decoder 400 not executing the
 20 method 800 all 32 subbands are determined, while only $sbl \leq 32$ subbands are calculated
 in accordance with the preferred method 800.

The computation required for the matrixing operation of the polyphase synthesis
 filterbank module 404 is expressed as:

$$V_i = \sum_{k=0}^{n-1} S_k \cos(\pi(2k+1)(n/2+i)/2n) \quad (3)$$

$i = 0, 1, \dots, 2n-1$ and $n = 32$.

In accordance with the method 800, Equation (3) becomes Equation (4) as follows:

$$V_i = \sum_{k=0}^{sb/2-1} S_k \cos(\pi(2k+1)(n/2+i)/2n) \quad (4)$$

5 where S_k is the k -th input coefficient for polyphase synthesis operations and V_i is the i -th output coefficient. Equation (4) shows the computational workload of the processor 105 implementing the method 800 decreases linearly with the bandwidth.

After the bitstream unpacking of step 802 (i.e., as performed by the Huffman decoding module 401), which require only a small percentage of the total computational workload (4% in our examples), the workload associated with the subsequent step 804 (i.e., as performed by the modules 402, 403 and 404) can be partitioned. A granularity may be selected that corresponds to all the 32 subbands defined in the MPEG 1 audio standard. However, for the sake of simplicity, in accordance with the preferred method 800, these 32 subbands are partitioned into only four groups, where each group
15 corresponds to a decoding level, as seen in Fig. 4 and Table 1.

As described above, the decoding Level 1 covers the lowest frequency bandwidth (0 - 5.5 kHz) which may be defined as the base layer. Although the base layer occupies only a quarter of the total bandwidth and contributes to roughly a quarter of the total computational workload performed by the processor 105 in decoding an audio clip, the
20 base layer is perceptually the most relevant frequency band. The output audio quality corresponding to Level 1 of Table 1 is certainly sufficient for services like news and sports commentary. Level 2 covers a bandwidth of 11 kHz and almost reaches the FM radio quality, which is sufficiently good even for listening to music clips, especially in noisy environments. Level 3 covers a bandwidth of 16.5 kHz and produces an output that

is very close to CD quality. Finally, Level 4 corresponds to the standard MP3 decoder, which decodes the full bandwidth of 22 kHz.

Levels 1, 2 and 3 process only a part of the data representing the different frequency components, whereas Level 4 processes all the data and is therefore computationally more expensive. The audio quality corresponding to levels 3 and 4 are almost indistinguishable in noisy environments, but are associated with substantially different power consumption levels.

Although each of the four frequency bands requires roughly the same workload, their perceptual contributions to the overall QoS are vastly different. In general, the low frequency band (i.e., Level 1) is significantly more important than any of the higher frequency bands.

The minimum operating frequency of the processor 105 for decoding audio data, in accordance with the method 800 at any particular decoding level, may be determined. The computed frequency can then be used to estimate the power consumption due to the processor 105. The variability in the number of bits constituting a granule and also the variability in the processor cycle requirement in processing any granule is taken into account. By accounting for this variability, the change in processor 105 frequency requirement when the playback delay of the portable computing device 100 is changed may be determined.

As described above and as seen in Fig. 5, the processor 105 uses the internal buffer 500 of size b , configured within memory 106, in decoding audio data in the form of an audio bit stream (e.g., an audio clip). The decoded audio stream, which is a sequence of PCM samples, is written into the playout buffer 501 of size B configured within memory 106. This playout buffer 501 is read by the processor 105 at some specified rate.

Assuming that the input bitstream to be decoded is fed into the internal buffer 500 at a constant rate of r bits/sec. The number of bits constituting a granule in the MP3 frame structure is variable. The maximum number of bits per granule can almost be three times the minimum number of bits in a granule, where this minimum number is around 1200 bits. To characterize this variability, two functions $\phi'(k)$ and $\phi''(k)$ may be used, where $\phi'(k)$ denotes the minimum number of bits constituting any k consecutive granules in an audio bitstream, and $\phi''(k)$ denotes the corresponding maximum number of bits. $\phi'(k)$ and $\phi''(k)$ can be obtained by analyzing a number of audio clips that are representative of audio clips to be processed.

Now, given an audio clip to be decoded, let $x(t)$ denote the number of granules arriving in the internal buffer 501 over the time interval $[0, t]$. Because of the variability in the number of bits constituting a granule, the function $x(t)$ will be audio clip dependent. Similar to the functions $\phi'(k)$ and $\phi''(k)$, two functions $\alpha'(\Delta)$ and $\alpha''(\Delta)$ to bound the variability in the arrival process of the granules into the internal buffer 501 may be used.

The two functions $\alpha'(\Delta)$ and $\alpha''(\Delta)$ are defined as follows:

$$\alpha'(\Delta) \leq x(t+\Delta) - x(t) \leq \alpha''(\Delta), x(t), \text{ and } t, \Delta \geq 0 \quad (5)$$

where $\alpha'(\Delta)$ denotes the minimum number of granules that can arrive in the internal buffer 501 within any time interval of length Δ , and $\alpha''(\Delta)$ denotes the corresponding maximum number.

Given the functions $\phi'(k)$ and $\phi''(k)$, it is possible to determine the pseudo-inverse of these two functions, denoted by $\phi'^{-1}(n)$ and $\phi''^{-1}(n)$, with the following interpretation. Both these functions take the number of bits n as an argument. $\phi'^{-1}(n)$ returns the maximum number of granules that can be constituted by n bits and $\phi''^{-1}(n)$ returns the

minimum number of granules that can be constituted by n bits. Since the input bit stream arrives in the internal buffer 501 at a constant rate of r bits/sec, $\alpha^l(\Delta)$ may be defined as follows:

$$\alpha^l(\Delta) = \varphi^{n-1}(r\Delta) \text{ and } \alpha^n(\Delta) = \varphi^{l-1}(r\Delta) \quad (6)$$

5 Again, since the number of processor cycles required to process any granule is also variable, this variability may be captured using two functions $\gamma^l(k)$ and $\gamma^n(k)$. Both the functions $\gamma^l(k)$ and $\gamma^n(k)$ take the number of granules k as an argument. $\gamma^l(k)$ returns the minimum number of processor cycles required to process any k consecutive granules and $\gamma^n(k)$ returns the corresponding maximum number of processor cycles. Fig. 6 shows
10 the cycle requirement for the processor 105 per granule, corresponding to a 160 kbits/sec bit rate audio clip, for a duration of around 30 secs. Fig. 6 shows the processor cycle requirement corresponding to the four decoding levels of Table 1. There are two points to be noted in Fig. 6: (i) the increasing processor cycle requirement as the decoding level is increased, (ii) the variability of the processor cycle requirement per granule for any
15 decoding level.

Assuming that the playout buffer 501 is readout by the processor 105 at a constant rate of c PCM samples/sec, after a playback delay (or buffering time) of d seconds. Usually c is equal to 44.1K PCM samples/sec for each channel (and therefore, $44.1K \times 2$ PCM samples/sec for stereo output) and d can be set to a value between 0.5 to 2 seconds.
20 If the number of PCM samples per granule is equal to s (which is equal to 576×2), the playout rate is equal to c/s granules/second. If the function $C(t)$ denotes the number of granules readout by the processor 105 over the time interval [0, t], then,

$$C(t) = \begin{cases} 0, & t \leq d \\ \frac{c}{s} \cdot t, & t > d \end{cases}$$

Now, given the input bitrate r , the functions $\phi^l(k)$, $\phi^u(k)$, $\gamma^l(k)$ and $\gamma^u(k)$ characterizing the possible set of audio clips to be decoded, and the function $C(t)$, the minimum processor frequency f to sustain the playout rate of c PCM samples/sec may be determined. This is equivalent to requiring that the playout buffer 501 never underflows.

- 5 If $y(t)$ denotes the total number of granules written into the playout buffer 501 over the time interval $[0, t]$, then this is equivalent to requiring that $y(t) \geq C(t)$ for all $t \geq 0$.

Let the service provided by the processor 105 at frequency f be represented by the function $\beta(\Delta)$. Similar to $\alpha^l(\Delta)$, $\beta(\Delta)$ represents the minimum number of granules that are guaranteed to be processed (if available in the internal buffer 500) within any time
 10 interval of length Δ . It may be shown that $y(t) \geq (\alpha^l \otimes \beta)(t)$, $t \geq 0$, where $\otimes$ is the min-plus convolution operator defined as follows.

For any two functions f and g , $(f \otimes g)(t) = \inf_{0 \leq s \leq t} \{f(t-s) + g(s)\}$. Hence, for the constraint $y(t) \geq C(t)$, $t \geq 0$ to hold, it is sufficient that the following inequality holds:

$$(\alpha^l \otimes \beta)(t) \geq C(t), \quad t \geq 0 \quad (7)$$

- 15 From the duality between $\otimes$ and $\oslash$, for any three functions f , g and h , $h \geq f \oslash g$ if and only if $g \otimes h \geq f$, where $\oslash$ is the min-plus deconvolution operator, defined as follows: $(f \oslash g)(t) = \sup_{s \geq 0} \{f(t+s) - g(s)\}$. Using this result on inequality (1), $\beta(t)$ may be determined as follows:

$$\beta(t) \geq (C \oslash \alpha^l)(t), \quad t \geq 0 \quad (8)$$

- 20 Note that $\beta(t)$ is defined in terms of the number of granules that need to be processed within any time interval of length t . To obtain the equivalent service in terms of processor cycles, the function $\gamma^u(k)$ defined above may be used. The minimum service that needs

to be guaranteed by the processor 105 to ensure that the playout buffer 501 never underflows is given by:

$$\bar{\beta}(t) = \gamma^u(\beta(t)) = \gamma^u((C \oslash \alpha^l)(t)) = \gamma^u(C(t) \oslash \varphi^{u-1}(rt)) \quad (9)$$

processor cycles for all $t \geq 0$. Hence, the minimum frequency at which the processor 105 should be run to sustain the specified playout rate is given by:
 $\min\{f \mid f \cdot t \geq \bar{\beta}(t), \forall t \geq 0\}$. The energy consumption while decoding an audio clip of duration t is proportional to $f^3 t$, assuming a voltage and frequency scalable processor, where corresponding to any operating point, the voltage is proportional to the clock frequency.

Fig. 7 shows the processor cycles required within any interval of length t corresponding to the decoding levels of Table 1. From Fig. 7, it can be seen that each decoding level is associated with a minimum (constant) frequency f . As the decoding level is increased, the associated value of f also increases.

Supposing the processor 105 is run at a constant frequency equal to f processor cycles/sec, corresponding to some decoding level. The minimum sizes of the internal and the playout buffers 500 and 501, which will guarantee that these buffers will never overflow, may be determined. The pseudoinverse of the two functions γ^l and γ^u , denoted by $\gamma^{l-1}(n)$ and $\gamma^{u-1}(n)$, respectively, may be determined. Both these functions γ^l and γ^u take the number of processor cycles n as an argument. $\gamma^{l-1}(n)$ returns the maximum number of granules that may be processed using n processor cycles and $\gamma^{u-1}(n)$ returns the corresponding minimum number.

The minimum number of granules that are guaranteed to be processed within any time interval of length Δ , when the processor 105 is run at a frequency f , is equal to

- 20 -

$\gamma^{u-1}(f\Delta)$. It may be shown that the minimum size b of the internal buffer 500, such that the internal buffer 500 never overflows is given by $b = \sup_{\Delta \geq 0} \{\alpha^u(\Delta) - \gamma^{u-1}(f\Delta)\}$ granules.

Similarly, the maximum number of granules that may be processed within any time interval of length Δ is given by $\gamma^{l-1}(f\Delta)$. It is possible to show that arrival process of granules in the playout buffer 501 is upper bounded by the function $\bar{\alpha}^u(\Delta)$, which may be determined as follows:

$$\bar{\alpha}^u(\Delta) = (\alpha^u(\Delta) \otimes \gamma^{l-1}(f\Delta)) \oslash \gamma^{u-1}(f\Delta), \quad \forall \Delta \geq 0 \quad (10)$$

where $\bar{\alpha}^u(\Delta)$ is the maximum number of granules that might be written into the playout buffer 501 within any time interval of length Δ . The minimum size of the buffer 501 (i.e., B), to guarantee that the buffer 501 never overflows can now be shown to be equal to $B = \sup_{\Delta \geq 0} \{\bar{\alpha}^u(\Delta) - C(\Delta)\}$ granules. The sizes b and B in terms of bits and PCM samples are $\varphi^u(b)$ and sB respectively.

In one implementation, the processor 105 may be an Intel XScale 400MHz processor with the decoding levels being set according to Table 2 below.

Table 2

Playback delay	Level 4	Level 3	Level 2	Level 1
0.5 sec	3.56 MHz	2.91 MHz	2.13 MHz	1.33 MHz
1.0 sec	3.32 MHz	2.71 MHz	1.99 MHz	1.23 MHz
2.0 sec	3.20 MHz	2.61 MHz	1.91 MHz	1.19 MHz

- 21 -

The aforementioned preferred method(s) comprise a particular control flow. There are many other variants of the preferred method(s) which use different control flows without departing the spirit or scope of the invention. Furthermore one or more of the steps of the preferred method(s) may be performed in parallel rather sequentially.

5

Industrial Applicability

It is apparent from the above that the arrangements described are applicable to the computer and data processing industries.

The foregoing describes only some embodiments of the present invention, and modifications and/or changes can be made thereto without departing from the scope and
10 spirit of the invention, the embodiments being illustrative and not restrictive.

(Australia Only) In the context of this specification, the word “comprising” means “including principally but not necessarily solely” or “having” or “including”, and not “consisting only of”. Variations of the word “comprising”, such as “comprise” and “comprises” have correspondingly varied meanings.

15

The claims defining the invention are as follows:

1. A method of decoding audio data representing an audio clip, said method comprising the steps of:

5 selecting one of a predetermined number of frequency bands;

decoding a portion of the audio data representing said audio clip according to the selected frequency band, wherein a remaining portion of the audio data representing said audio clip is discarded; and

10 converting the decoded portion of audio data into sample data representing the decoded audio data.

2. The method according to claim 1, further comprising the step of partitioning the frequency range of the audio data representing said audio clip into said frequency bands.

15 3. The method according to claim 1, wherein each of said frequency bands is associated with a different level of power consumption for a portable audio device.

4. The method according to claim 1, wherein the audio data is an MP3 bitstream.

20 5. A decoder for decoding audio data representing an audio clip, said method comprising the steps of:

decoding level selection means for selecting one of a predetermined number of frequency bands;

25 decoding means for decoding a portion of the audio data representing said audio clip according to the selected frequency band, wherein a remaining portion of the audio data representing said audio clip is discarded; and

data conversion means for converting the decoded portion of audio data into sample data representing the decoded audio data.

6. A portable electronic device comprising:

5 decoding level selection means for selecting one of a predetermined number of frequency bands;

decoding means for decoding a portion of audio data representing an audio clip according to the selected frequency band, wherein a remaining portion of the audio data representing said audio clip is discarded; and

10 data conversion means for converting the decoded portion of audio data into sample data representing the decoded audio data.